# Метод разработки документации семейств программных продуктов

К.Ю.Романовский
kr@tepkom.ru

Санкт—Петербургский государственный университет
198504, Университетский пр., 28
Санкт—Петербург, Россия

В работе представлен новый метод разработки документации семейств программных продуктов. Метод включает в себя процесс, средства визуального проектирования структуры и возможностей повторного использования документации семейства и отдельных продуктов, язык DRL (Documentation Reuse Language), предназначенный для разметки текстов документации, а также архитектуру инструментальных средств для поддержки метода. Реализован прототип пакета инструментальных средств. Метод апробирован для создания документации в проекте разработки ПО семейства систем управления вещанием.

## Введение

Разработка семейств программных продуктов – популярный подход к разработке ПО, позволяющий эффективно реализовать повторное использование различных активов процесса разработки ПО [7, 6, 12]. Семейство программных продуктов – это набор совместно разрабатываемых программных продуктов, относящихся к одной предметной области и разделяющих общую функциональность. Традиционные методы разработки семейств программных продуктов поддерживают повторное использование разнообразных общих активов – программных компонент, архитектуры, тестовых сценариев, требований и даже языков, ориентированных на предметную область (Domain—Specific

Languages) [4, 19, 18, 15]. В ряде работ предлагается использовать визуальные модели для облегчения управления общими активами – это диаграммы возможностей (Feature Diagrams) [8] и UML [2]. Однако за рамками рассмотрения остается техническая документация, разработка и сопровождение которой также является трудоемким процессом – практически всегда, когда выполняются изменения кода необходимо соответствующим образом обновить и документацию. Вместе с тем, техническая документация различных продуктов семейства имеет много общего.

В области разработки технической документации популярен подход единого исходного представления (Single Sourcing), который обеспечивает разделение содержания документации и форматирования [24, 21, 5]. Этот подход предоставляет возможность на основе единого исходного представления получить комплект документов, таких как руководство пользователя, справочная система, руководство по быстрому старту и т.п. При этом документы могут быть представлены в различных форматах – HTML, PDF, HTMLHelp, Microsoft Word и др. Также известен и используется подход DITA (Darwin Information Typing Architecture), предложенный компанией IBM [9, 14]. В DITA документацию предлагается строить в виде набора топиков (topic), каждый из которых можно использовать в любом контексте в произвольном документе. Однако эти подходы не учитывают особенностей процесса разработки документации семейств программных продуктов, в частности того факта, что повторно используемые фрагменты зачастую требуют адаптации к использованию в различных документах (такое повторное использование называется вариативным). Фактически, единственное средство, обеспечивающее вариативное повторное использование в указанных подходах – это условное включение фрагментов текста. Не поддерживается также и визуальное моделирование для проектирования структуры пакетов документации.

В работе предлагается метод разработки технической документации семейств программных продуктов, который включает в себя:

- Модельно—ориентированный процесс разработки документации, согласованный с процессом разработки семейств программных продуктов [12];
- Визуальную модель документации, предназначенную для проектирования пакетов документов и структуры повторного использования документации и основанную на модели возможностей, применяемой при разработке семейств программных продуктов [8];
- Язык разметки документации DRL (Documentation Reuse

Language), обеспечивающий вариативное повторное использование документации и использующий идеи DocBook [24] для реализации принципов единого исходного представления, а также концепцию фреймов Бассета [3] для параметризации крупных фрагментов текста;

- Поддержку циклической разработки (round—trip) модели и текстов документации;
- Архитектуру пакета инструментальных средств, поддерживающих использование метода в процессе разработки семейств программных продуктов.

# 1 Обзор литературы

В данной работе затрагиваются следующие области разработки ПО: методы разработки семейств программных продуктов, визуальное моделирование в разработке семейств программных продуктов, средства разработки технической документации.

## 1.1 Методы разработки семейств программных продуктов

Основная идея, лежащая в основе большинства методов разработки семейств программных продуктов, и впервые предложенная в методе FODA [17], заключается в явном разделении всего процесса на две части – разработку общих активов (domain engineering) и разработку конкретных продуктов семейства (application engineering) [17, 4, 19, 15].

На взгляд автора, наиболее удачное описание процесса разработки семейств программных продуктов приводится в работе [12]: выделяются две составляющие процесса – разработка семейства продуктов (аналог domain engineering в FODA) и разработка продуктов (аналог application engineering в FODA). Разработка семейства, в основном, выполняется в начале процесса разработки, до массового выпуска продуктов. Тем не менее, допускается возврат к разработке семейства из процесса разработки продуктов, поскольку при добавлении новых продуктов может потребоваться модификация активов семейства.

Процесс разработки семейства по [12] изображен на Рис. 1.

На этапе анализа семейства необходимо определить, какие продукты могут разрабатываться в рамках семейства, а также выделить общие и вариативные свойства продуктов семейства.

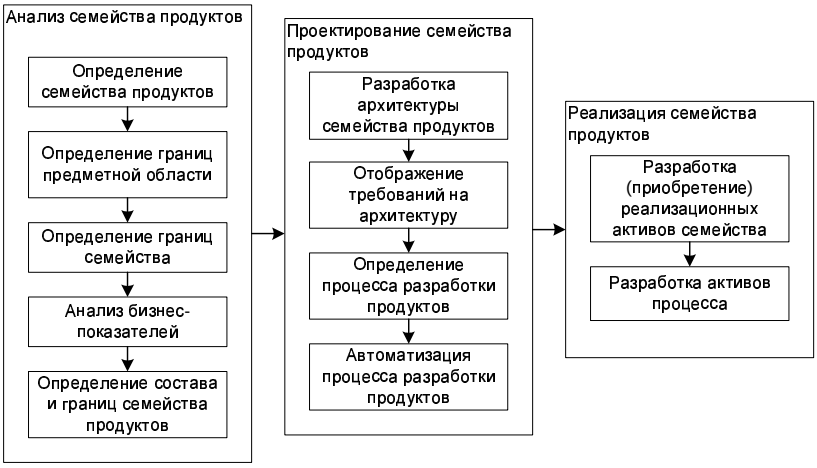


Рис. 1: Процесс разработки семейства.

На этапе проектирования семейства нужно решить, как будет разрабатываться семейство и продукты, разработать архитектуру семейства, спроектировать переиспользуемые активы, определить процесс разработки продуктов семейства и спроектировать автоматизацию этого процесса.

Основные задачи этапа реализации семейства – реализовать общие активы семейства, в том числе активы процесса в соответствии с планом автоматизации процесса разработки продуктов.

Процесс разработки продукта в рамках семейства изображен на Рис. 2.

Этап анализа задачи необходим для выяснения того, действительно ли поставленная задача может быть решена в рамках разрабатываемого семейства, а также какие именно части задачи могут быть решены в рамках семейства, а какие требуют дополнительной работы и интеграции с семейством.

Основная задача этапа спецификации продукта – определить и формализовать требования к разрабатываемому продукту, а также описать, как именно продукт будет разрабатываться в рамках семейства – какие общие активы могут быть использованы и как потребуется их настроить.

Основная задача этапа разработки вспомогательных материалов – выпустить все необходимые активы продукта, не имеющие отношения к исполняемому коду, в частности, маркетинговые материалы, докумен-

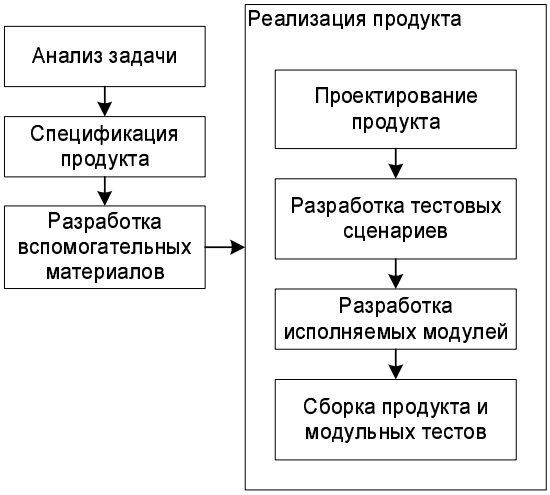


Рис. 2: Процесс разработки продукта в рамках семейства.

тацию, упаковку и т.п.

Главная задача этапа реализации продукта – разработать требуемый продукт.

## 1.2 Визуальное моделирование в разработке семейств программных продуктов

При анализе и проектировании семейств программных продуктов нередко используются языки и средства визуального моделирования: UML [2], модель возможностей (feature model) [17, 19, 8, 11], языки, ориентированные на предметную область (DSL) [23]. Основной источник проблем, возникающих при использовании визуального моделирования – необходимость отображать все возможные различия продуктов семейства (т.е. вариативность семейства). В большинстве подходов интеграция вариативности в модели и визуальный язык приводит к тому, что модели и диаграммы становятся громоздкими и бесполезными.

Наиболее удачно проблема моделирования вариативности семейства продуктов решена в модели возможностей, впервые предложенной в работе [17]. Ключевое понятие этой модели – возможность (feature), которая является обособленным свойством системы, распознаваемым с точки зрения пользователя или разработчика. Модель возможностей – это набор возможностей и иерархических связей между ними

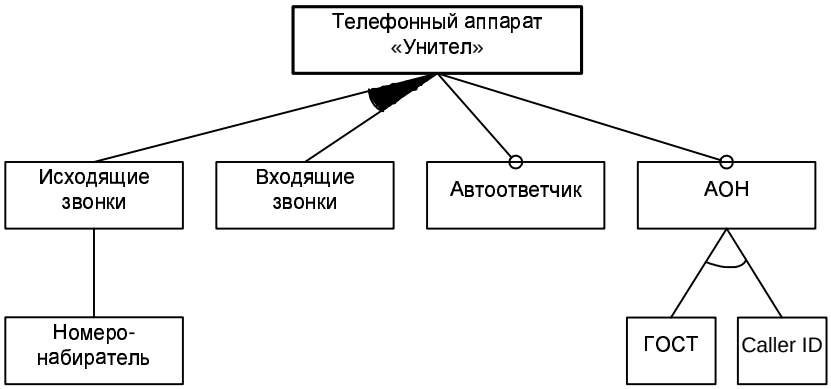


Рис. 3: Диаграмма возможностей.

с явно выделенным корнем иерархии, который называется концептом (concept) [8]. Иерархическая связь отображает декомпозицию концепта или возможности и называется отношением включения. Модель возможностей строится в основном на этапе анализа семейства программных продуктов и характеризует семейство в целом.

Основная задача модели возможностей – формализовать общие и различные свойства продуктов семейства. Для этого применяются различные разновидности отношения включения.

Для изображения модели возможностей используются диаграммы возможностей. Фрагмент диаграммы возможностей в классической нотации [8] изображен на Рис. 3.

Диаграмма на Рис. 3 описывает семейство телефонных аппаратов “Унител”, в которых могут быть реализованы исходящие звонки, входящие звонки, автоответчик и АОН. Прямоугольником изображается концепт или возможность. Надпись, содержащаяся внутри прямоугольника – это имя соответствующей сущности. Линия, соединяющая концепт с возможностью либо две возможности между собой, изображает обязательное включение (возможность “исходящие звонки” требует наличия возможности “Номеронабиратель”), либо необязательное включение, если линия помечена кружочком со стороны подвозможности (телефонный аппарат может содержать возможность “Автоответчик”).

На практике телефонный аппарат, который не может обслуживать ни исходящие, ни входящие вызовы лишен смысла, так что, по крайней мере, один из этих элементов должен быть представлен в каждом продукте семейства. Для выражения этого факта используется ограниче-

ние “произвольное включение”, для изображения которого на диаграмме линии отношений, на которые накладывается ограничение, соединяются дугой и угол, ограничиваемый этой дугой, закрашивается (группа включений возможностей “Исходящие звонки” и “Входящие звонки” на Рис. 3).

В ситуации, когда в каждом продукте может быть представлена не более чем одна возможность, используется ограничение <альтернативное включение>. Для обозначения альтернативного включения линии отношений, на которые накладывается ограничение, соединяются дугой. На Рис. 3 альтернативное включение используется для выражения того факта, что АОН может быть либо российским (ГОСТ), либо европейским (Caller ID).

Любое отношение включения, кроме обязательного включения, является точкой вариации. Точка вариации – это позиция в модели возможностей, которая описывает варианты различной компоновки возможностей в составе различных продуктов семейства.

В различных методах разработки семейств программных продуктов модель возможностей играет разную роль [8]. Например, она может применяться для конфигурирования продуктов семейства, как предлагается в методе FeaturSEB [13], или как основная модель семейства, вокруг которой интегрируется весь процесс разработки, как в методе DEMRAL [20].

## 1.3 Средства разработки технической документации

Для разработки технической документации используются различные технологии – это и редакторы общего назначения, такие как [21], и специализированные системы [25], и “легкие” системы на основе XML и открытых стандартов [24]. Также в этой области вполне возможно использование систем управления контентом, таких как Arbortext. Рассмотрим указанные технологии создания документации с точки зрения возможностей повторного использования.

**Переиспользование не поддерживается.** К этому классу относятся такие системы, как Microsoft Word[1], Adobe PageMaker[2], TeX[3]. Системы этого класса, как правило, предназначены для разработки одиночных документов, либо небольших пакетов документов. Повторное использование документации возможно только на уровне цельных документов или глав.

**Крупноблочное повторное использование с помощью механизма включения.** Типичные представители этого класса технологий разработки документации – Adobe FrameMaker [21] и DocBook [24]. Системы этого класса ориентированны на разработку пакетов документов. Основной механизм переиспользования – включение крупных фрагментов текста в различные контексты. Основной механизм параметризации повторно используемого текста – условное включение, т.е. применение внешнего условия, которое управляет тем, попадет ли данный фрагмент текста в данный документ. Как правило, такой уровень параметризации достаточен для описания небольших отличий в документации, таких как опции, которые могут отсутствовать в различных вариантах поставки системы, либо на различных целевых платформах.

**Явное выделение повторно используемого контента.** В этом классе наиболее популярны технологии DITA [9] и AuthorIT [25]. Системы этого уровня ориентированны на изначальное планирование переиспользования. Документация изначально представляет собой набор элементов, которые могут быть использованы в различных контекстах. Однако возможности по параметризации по—прежнему ограничиваются условным включением.

**Параметризация повторно используемых элементов.** Примеры технологий этого класса – Netron Fusion [22] и XVCL [15], реализующие идею фреймов Бассета [3]. Системы подобного класса предоставляют возможность параметризовать переиспользуемые фрагменты в зависимости от потребностей контекста, в котором они используются. В частности, можно добавлять текст в заранее определенные места, а также

[1]Коммерческий текстовый редактор общего назначения, разрабатываемый компанией Microsoft (http://www.microsoft.com/word/).

[2]Коммерческая издательская система компании Adobe для верстки небольших печатных изданий, таких как брошюры и газеты (http://www.adobe.com/products/pagemaker/index.html).

[3]Свободно-распространяемая издательская система, разработанная Дональдом Кнутом и изначально предназначенная для создания математических текстов. Стала стандартом де-факто для оформления научных публикаций.

опускать те или иные фрагменты текста. Механизмы параметризации в системах этого класса хорошо подходят для крупных повторно используемых фрагментов, однако слишком громоздки для реализации мелкозернистого повторного использования с незначительными модификациями текста, такими как изменение синтаксической формы слов.

**Информационное моделирование, планирование повторного использования.** К этому классу можно отнести такие технологии, как DITA и Arbortext[4]. Системы этого класса ориентированы на использование контента в большом количестве различных контекстов. При этом средства параметризации достаточно бедны (условное включение), однако довольно развиты средства описания контекста, выбора контента для использования вплоть до наличия языка запросов, позволяющего подключить в тот или иной контекст все элементы, удовлетворяющие заданному условию.

# 2 Метод

Предлагаемый метод состоит из следующих частей:

- Процесс разработки документации;
- Визуальная модель документации семейства и отдельных продуктов;
- Язык разметки документации DRL (Documentation Reuse Language);
- Архитектура пакета инструментальных средств.

## 2.1 Процесс

Процесс разработки документации, как и процесс разработки семейства в целом [12], состоит из двух частей – разработки документации семейства и разработки документации продуктов. При необходимости допускается вносить изменения в документацию семейства при разработке документации продуктов – это может потребоваться для тонкой настройки повторного использования документации.

Рассмотрим указанные части подробнее.

[4]Промышленная система управления контентом (http://www.ptc.com/appserver/mkt/products/home.jsp?k=3591).

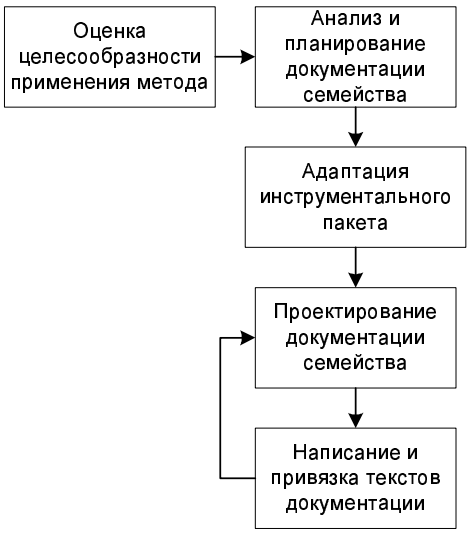


Рис. 4: Схема процесса разработки документации семейства.

### 2.1.1 Разработка документации семейства

Процесс разработки документации семейства схематически изображен на Рис. 4.

Перед использованием метода необходимо оценить целесообразность его применения. В контексте процесса разработки семейств программных продуктов, описанного в работе [12] этот шаг выполняется во время анализа семейства продуктов. Метод применим практически в любых условиях, однако в ряде случаев его использование дает существенные преимущества по сравнению с традиционными методами разработки документации:

1. Документация различных продуктов семейства имеет много общего, при этом переиспользуемые фрагменты имеют различия. В случае отсутствия различий вполне эффективно будут работать традиционные методы разработки документации.

2. Регулярно разрабатываются новые версии существующих продуктов или новые продукты семейства. Если же документация разрабатывается один раз и исправления в нее вноситься не должны либо очень редки, то традиционные методы разработки документации вполне удовлетворят потребности разработчика.

3. Ошибки в документации недопустимы (например, в случае mission—critical систем). Если же документация некритична, и возможно допускать в ней ошибки и несвоевременно обновлять ее с выходом новых версий, то возможно дешевле будет использовать простейший метод повторного использования – обычное копирование нужных фрагментов текста.

Использование метода требует обучения технических писателей, причем для тонкой настройки повторного использования от технического писателя потребуется навык написания XML—документов [1]. Для технических писателей, имеющих опыт работы с такими технологиями как DITA и DocBook, обучение не составит большого труда, поскольку они также основаны на редактировании XML— документов. Кроме того, организация, применяющая предлагаемый метод, должна быть достаточно стабильной и иметь долгосрочные планы, связанные с разработкой семейства программных продуктов, поскольку в краткосрочной перспективе стоимость документации возрастает, как и в случае с любым методом планируемого повторного использования общих активов. Таким образом, перед принятием решения об использовании метода, необходимо оценить также и экономическую целесообразность его применения.

После принятия решения об использовании метода, необходимо выполнить анализ потребностей пользователей и планирование состава документации. Этот этап также выполняется при разработке семейства продуктов, а именно во время анализа семейства. На этом этапе определяется, какие документы должны входить в состав документации (руководство пользователя, справочная система, краткий справочник, и т.п.). Полученный список документов ложится в основу модели документации.

За анализом и планированием следует работа по адаптации инструментального пакета. Это необходимо для интеграции инструментальных средств разработки документации в единую среду разработки семейства. В большинстве случаев адаптация будет ограничиваться конфигурированием инструментальных средств, однако в ряде случаев может потребоваться доработка отдельных инструментов с учетом особенностей конкретного семейства и специфических потребностей пользователей. Адаптация инструментального пакета начинается при проектировании семейства (на этапах определения и автоматизации процесса разработки) и завершается при реализации семейства (на этапе разработки активов процесса).

Далее выполняется проектирование документации семейства. Цель этого этапа – выявить и формализовать возможности повторного ис-

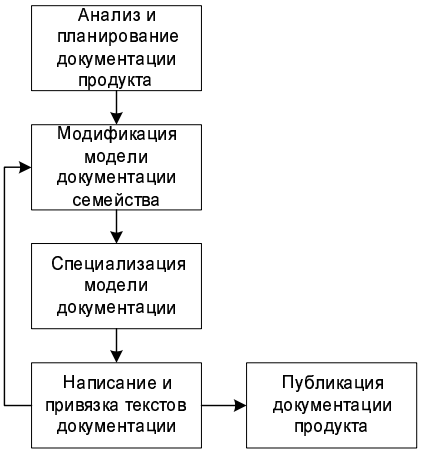


Рис. 5: Схема процесса разработки документации продукта.

пользования документации в рамках семейства. В традиционном процессе разработки семейств программных продуктов этот этап выполняется во время проектирования семейства. Для формализации результатов проектирования документации используется высокоуровневая визуальная модель документации. В процессе разработки продуктов семейства возможен возврат к этапу проектирования документации, в случае, когда обнаруживаются новые возможности повторного использования. Также возврат к этапам анализа и проектирования документации необходим при добавлении новых продуктов в семейство.

Написание и привязка текстов документации выполняется частично при разработке семейства, а именно во время реализации семейства. Целью этого этапа является наполнение каркаса документации семейства содержанием, а также привязка документации к модели.

### 2.1.2 Разработка документации продукта

Процесс разработки документации продукта семейства схематически изображен на Рис. 5.

Задача этапа анализа и планирования документации – выявить требования к документации продукта семейства, проанализировать возможности повторного использования существующей документации, спланировать работу по доработке документации. В контексте процесса разработки семейств анализ и планирование документации продукта выполняется во время разработки продукта, а именно на этапе анализа

задачи.

Модификация модели необходима для добавления в модель недостающих информационных элементов, специфичных для нового продукта. В контексте процесса разработки семейства модификация модели выполняется во время разработки продукта на этапе спецификации продукта.

Специализация модели документации также выполняется на этапе спецификации продукта. Цель специализации модели – выбрать во всех точках вариации конкретный набор включаемых возможностей.

Написание и привязка текстов документации продукта, по сути, ничем не отличается от аналогичного процесса в рамках разработки семейства. Этот процесс выполняется во время разработки продукта на этапе разработки вспомогательных материалов.

Публикация документации – это порождение конечных документов на основе информации в модели и в текстах документации. Процесс публикации документации выполняется на этапе сборки продукта и модульных тестов. Кроме того, публикация документации может выполняться и раньше, например, в целях проверки качества документации.

Последний этап в процессе разработки документации продукта – публикация документации. Публикация документации выполняется на этапе сборки продукта и модульных тестов, а также в процессе написания текстов для проверки правильности текстов.

### 2.1.3 Циклическая разработка модели и текстов документации

После того, как выполнен первоначальный цикл разработки документации в соответствии с предложенным процессом, разработчик имеет следующие представления документации: модель документации, исходные тексты документации, а также итоговые документы (которые автоматически генерируются по исходным текстам). Стрелками на Рис. 6 обозначены возможные направления внесения изменений.

Основное направление распространения изменений – от модели к текстам, поскольку на первых этапах разработки документации как семейства в целом, так и конкретного продукта, выполняется анализ и проектирование документации, а также модификация модели в случае разработки документации продукта. Однако при написании текстов могут быть выявлены незамеченные ранее возможности повторного использования, которые удобно формализовать непосредственно в тексте документации, а впоследствии автоматически перенести в модель. Предлагаемый автором процесс допускает такой способ распространения изменений.

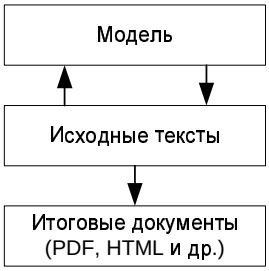


Рис. 6: Уровни представления документации.

## 2.2 Модель

Модель документации состоит из двух частей – модели документации семейства и модели документации продукта.

### 2.2.1 Модель документации семейства

Модель документации семейства предназначена для описания структуры документации и формализации возможностей повторного использования. Модель документации основывается на модели возможностей. Основная сущность модели документации – информационный продукт.

Информационный продукт – это документ или пакет документов, который является самостоятельным результатом процесса разработки документации. Примеры информационных продуктов – краткий справочник, руководство пользователя, справочная система и т.п. Информационные продукты могут включать в себя произвольно количество информационных элементов.

Информационный элемент – это фрагмент документации, предназначенный для использования в одном или нескольких информационных продуктах. Информационные элементы также могут включать в себя другие информационные элементы.

Семантически информационный продукт соответствует концепту в модели возможностей, а информационный элемент соответствует возможности.

Обе разновидности отношения включения и все ограничения, которые используются в модели возможностей, также используются в модели документации.

Нотация модели документации повторяет нотацию модели возможностей с той лишь разницей, что информационный элемент обозначается прямоугольником с двойными границами. Пример диаграммы до-

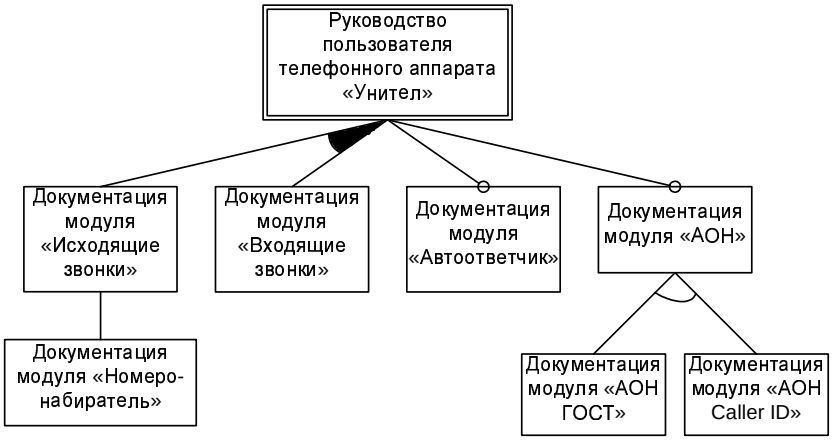


Рис. 7: Диаграмма документации семейства телефонных аппаратов “Унител”.

кументации для телефонного аппарата “Унител” изображен на Рис. 7.

### 2.2.2 Модель документации продукта

Модель документации продукта предназначена для формализации состава документации конкретного продукта. Сущности модели документации продукта такие же, как и в модели документации семейства. Однако в модели документации продукта допускается единственный вид отношений – обязательное включение. Модель документации продукта строится во время разработки документации продукта на этапе специализации модели документации. Построение модели документации продукта выполняется путем выбора необходимых информационных элементов во всех точках вариации модели документации семейства. Пример модели документации продукта для семейства телефонных аппаратов “Унител” показан на Рис. 8.

Модель документации на Рис. 8 выполнена для одного из продуктов семейства телефонных аппаратов “Унител”, а именно для телефонного аппарата, поддерживающего только входящую связь с возможностью определения номера вызывающего абонента по протоколу CallerID.

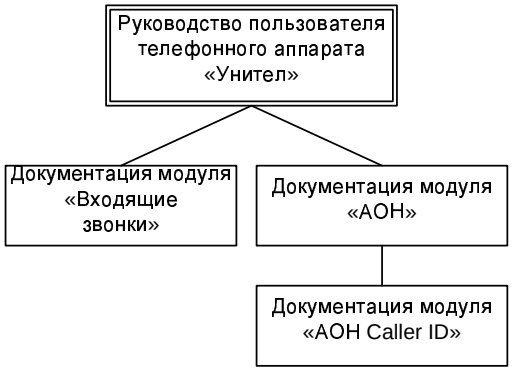


Рис. 8: Диаграмма документации продукта.

## 2.3 Язык DRL

Язык разметки документации DRL (Documentation Reuse Language) служит для описания документации семейства программных продуктов и документации отдельных продуктов. В данном разделе описываются основные конструкции языка, их семантика и взаимосвязи. Все конструкции разбиваются на следующие группы:

- Структура документации семейства;
- “Мелкозернистое” повторное использование;
- Документация продукта;
- Вспомогательные конструкции (условный блок и присваивание значений псевдопеременных);
- Форматирование текста.

Для обработки текстов традиционно используется технология XML [26]. Синтаксис языка DRL основывается на технологии XML. Каждая конструкция представляется XML—тегом. Для общего представления о синтаксисе языка приводятся листинги с фрагментами документации, на которые даются ссылки при описании семантики различных конструкций.

Ряд конструкций языка DRL основывается на хорошо зарекомендовавших себя конструкциях существующих языков. Вспомогательные конструкции основаны на конструкциях инструментального пакета FrameMaker [21]. Конструкции форматирования текста позаимствованы

из языка DocBook [24]. Новыми являются конструкции, обеспечивающие "мелкозернистое" переиспользование текста. Также новой является конструкция "Адаптер", разработанная на основе конструкции COPY фреймов Бассета [3] и конструкции adapt технологии XVCL [15]. Основное отличие конструкции "Адаптер" от указанных конструкций – это возможность разделения описания семейства документов и специализации этих документов под требования конкретного продукта. Оригинальные конструкции COPY и adapt одновременно подключали и специализировали переиспользуемый фрагмент. В языке DRL подключение переиспользуемого фрагмента осуществляется отдельной конструкцией ("включение информационного элемента"), а специализировать переиспользуемый фрагмент можно с помощью конструкции "Адаптер" непосредственно в процессе разработки документации конкретного продукта, не затрагивая документацию семейства.

Рассмотрим подробно основные конструкции языка DRL.

### 2.3.1 Структура документации семейства

Язык DRL тесно связан с моделью документации. Конструкции языка DRL, описывающие структуру документации, соответствуют основным сущностям модели:

- Информационный продукт;
- Информационный элемент;
- Включение информационного элемента.

Документация семейства описывает всевозможные документы, которые должны быть разработаны для различных продуктов семейства и повторно используемые фрагменты текста. Документы описываются с помощью конструкции "Информационный продукт". Примеры информационных продуктов – руководство пользователя, справочная система, учебные материалы. Фактически, информационный продукт описывает не единственный документ, а семейство документов, например семейство руководств пользователя для различных продуктов. Информационный продукт (см. Листинг 1, тег <infproduct />) содержит текст на языке DocBook (в примере Листинг 1 это теги <book /> и <bookinfo />), в который вставлены конструкции управления повторным использованием: включение информационного элемента, включение элемента словаря, включение элемента каталога. Рассмотрим эти конструкции подробнее.

Информационный элемент соответствует одноименной сущности модели документации семейства, т.е. является фрагментом текста, подготовленным к использованию в одном или нескольких информационных продуктах. Информационные элементы обеспечивают крупноблочное повторное использование. Помимо текста на языке DocBook информационный элемент может содержать конструкции управления повторным использованием (включение информационного элемента, включение элемента словаря, включение элемента каталога), а также конструкции, обеспечивающие вариативность, т.е. возможность адаптировать информационный элемент к специфическим особенностям различных контекстов использования. Вариативность достигается за счет параметров, привязанных к информационному элементу, а также за счет точек расширения, расположенных в тексте информационного элемента (Листинг 4, тег <nest />). Задание значений параметров и операции с точками расширения осуществляется в документации конкретных продуктов. Таким образом, в документации семейства включение информационного элемента описывается с помощью ссылки на него. Листинг 1 содержит пример информационного элемента (тег <infelem />) и включения информационного элемента (тег <infelemref />).

```
<infproduct title="Руководство пользователя">
  <book>
    <bookinfo>
    <title>Комплекс автоматизации вещания.</title>
    </bookinfo>
    <infelemref id="server" title="Серверная подсистема"/>
    <conditional condition="maintainable=yes">
         <infelemref id="maintenance"
                     title="Обслуживание сервера"/>
    </conditional>
  </book>
</infproduct>

<infelem title="Серверная подсистема">
...
</infelem>
```

Листинг 1: Информационный продукт и информационный элемент.

### 2.3.2 “Мелкозернистое” повторное использование

В данную группу входят следующие конструкции, которые обеспечивают повторное использование небольших фрагментов текста, вплоть до отдельных слов:

- Словарь;
- Включение элемента словаря;
- Каталог;
- Шаблон отображения элемента каталога;
- Включение элемента каталога.

Словарь – это набор пар имя—значение. Словари удобно использовать для описания стандартных терминов. Включение элемента словаря специфицируется путем указания конкретного словаря и имени элемента. При выполнении публикации документации ссылки по имени на элементы словаря будут замещаться на значения соответствующих элементов словаря. Пример словаря и включения элемента словаря приведен в Листинг 2 – это, соответственно, теги <dictionary /> и <dictentry />.

```
<dictionary name="terms">
  <entry name="ProductName">Видеосервер</entry>
  <entry name="ProductVersion">1.0</entry>
</dictionary>

<infelem title="About">
  О программе <dictentry dict="terms" name="ProductName"/>
</infelem>
```

Листинг 2: Словарь.

Каталог – это конструкция, задающая набор данных сходной структуры, которые могут представляться в итоговом тексте в различных видах (см. Листинг 3, тег <directory />). Типичный пример каталога – набор команд приложения (сохранение файла, текстовый поиск, печать и т.п.). Каждая команда может иметь название, описание, комбинацию клавиш, текст всплывающей подсказки, пиктограмму. В разных контекстах (описание меню, панели инструментов, описание возможностей приложения и т.п.) может потребоваться предоставить различную информацию о команде. Для этого предназначена конструкция <шаблон отображения элемента каталога>, описывающая какие параметры

элемента и в каком формате нужно отображать (см. Листинг 3, тег <dirtemplate />). Для каждого каталога может быть задано произвольное количество шаблонов отображения. Для включения элемента каталога необходимо указать имя шаблона и имя элемента каталога (см. Листинг 3, тег <direntry />). При выполнении публикации документации конструкция включения элемента каталога замещается на значение соответствующего элемента каталога, отформатированное в соответствии с заданным шаблоном.

```
<directory name="commands">
  <entry name="About">
    <attr name="name">О программе</attr>
           <attr name="hint">Информация о программе</attr>
           <attr name="icon">about.jpg</attr>
  </entry>
  <entry name="Exit">
    <attr name="name">Выход</attr>
           <attr name="hint">Выход из программы</attr>
           <attr name="icon">exit.jpg</attr>
  </entry>
</directory>

<dirtemplate name="CommandNameHint" directory="commands">
  <attrref name="name"/> (<attrref name="hint"/>)
</dirtemplate>

<dirtemplate name="CommandNameIcon" directory="commands">
  <attrref name="name"/>
  <image><fileref><attrref name="icon"/></fileref></image>
</dirtemplate>

<infelem title="Command List">
   Команды: <direntry templ="CommandNameIcon" name="About"/>,
            <direntry templ="CommandNameIcon" name="Exit"/>
</infelem>
```

Листинг 3: Каталог.

Словари и каталоги обеспечивают мелкозернистое повторное использование. Шаблоны отображения элементов каталога обеспечивают вариативность мелкозернистого повторного использования. В частности, задавая в атрибутах элемента каталога различные словоформы од-

ного и того же выражения (различные падежи, единственное и множественное число, заглавная буква) и выбирая соответствующий шаблон отображения, можно организовать повторное использование выражения с возможностью выбора конкретной словоформы.

### 2.3.3 Документация продукта

Документация продукта – это подмножество документации семейства с добавлением фрагментов текста, специфичных для данного продукта. Данная группа включает следующие конструкции:

- Специализированный информационный продукт;
- Адаптер.

Специализированный информационный продукт соответствует информационному продукту в документации семейства и описывает, какие информационные элементы должны войти в документацию данного продукта, а также как их необходимо настроить. Для настройки включаемых информационных элементов используется конструкция <адаптер>. В терминах модели документации продукта, адаптер привязывается к отношению включения и специфицирует значения доступных параметров включаемого элемента, а также при необходимости выполняет модификацию точек расширения. Поддерживаются следующие варианты модификации точек расширения: вставка текста до или после точки расширения, а также удаление содержимого точки расширения. Листинг 4 показывает пример использования конструкций документации продукта: тег <finalinfproduct /> соответствует специализированному информационному элементу, а тег <adapter /> – адаптеру.

```
<finalinfproduct title="Руководство пользователя">
   <setvalue name="maintainable">no</setvalue>
   <adapter infelemrefid="server">
    <insert---before nest="Mount">
      монтируется в стойку 19 дюймов
    </insert---before>
   </adapter>
</finalinfproduct>

<infproduct title="Руководство пользователя">
  <book>
    <bookinfo>
    <title>Комплекс автоматизации вещания.</title>
```

```
    </bookinfo>
    <infelemref id="server" title="Серверная подсистема"/>
    <conditional condition="maintainable=yes">
         <infelemref id="maintenance"
                     title="Обслуживание сервера"/>
    </conditional>
  </book>
</infproduct>

<infelem title="Серверная подсистема">
  Тип монтажа оборудования: <nest name="Mount"/>
</infelem>
```

Листинг 4: Специализированный информационный продукт и адаптер.

### 2.3.4 Вспомогательные конструкции

Группа вспомогательных конструкций содержит следующие конструкции:

- Условный блок;
- Присваивание псевдопеременной.

Условный блок (Листинг 1, тег <conditional />) может использоваться в тексте информационных продуктов и элементов. Эта конструкция содержит условие (в котором могут фигурировать значения параметров информационного элемента, а также псевдопеременных). При выполнении публикации документации условие проверяется, и если оно выполнено, то содержимое условного блока включается в итоговый документ.

Конструкция присваивания псевдопеременной (см. Листинг 4, тег <setvalue />) может использоваться в тексте информационных продуктов и элементов, а также в специализированном информационном продукте. Присвоенное значение может использоваться при вычислении параметров информационного элемента, а также в условии условного блока.

### 2.3.5 Форматирование текста

Для того чтобы разрабатываемая документация была качественно оформлена, язык должен включать конструкции, обеспечивающие описание смысловой структуры документов (главы, разделы, параграфы),

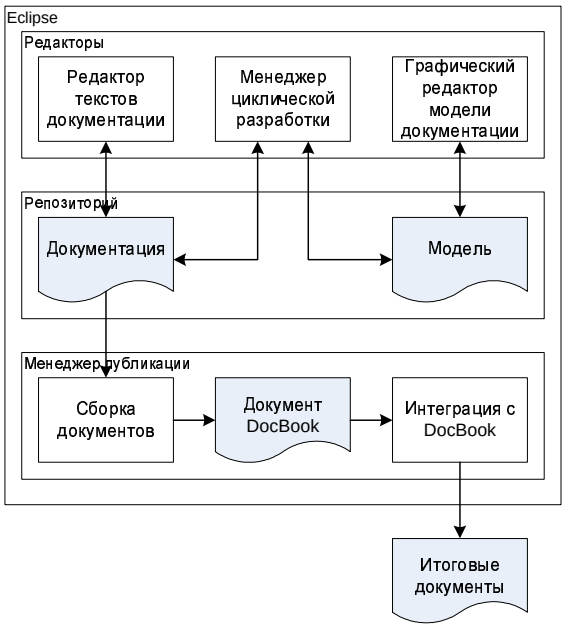


Рис. 9: Архитектура пакета инструментальных средств.

разметку текста (заголовки, выделение шрифтами, оформление списков, таблиц), оформление рисунков, а также ряд вспомогательных конструкций форматирования текста. Автор предлагает для этой цели использовать язык разметки, предложенный в технологии DocBook [24]. Таким образом, текстовое наполнение документов выполняется на языке DocBook.

## 2.4 Архитектура пакета инструментальных средств

Для использования метода автор предлагает реализовать пакет инструментальных средств, обеспечивающих редактирование модели и документации, циклическую связь модели и документации, а также публикацию итоговых документов. Архитектура пакета инструментальных средств представлена на Рис. 9. Все инструментальные средства интегрируются в единую среду разработки документации. Оптимальным вариантом является реализация всех инструментальных средств в виде модулей расширения популярной среды разработки Eclipse [10].

Графический редактор модели документации обеспечивает создание

и редактирование модели документации. Задача редактора текстов документации – обеспечивать создание и редактирование текстов документации. Менеджер циклической разработки интегрирует эти редакторы и обеспечивает пользователю интуитивно понятный способ циклической разработки, т.е. автоматизированное распространение изменений модели на документацию и наоборот.

Документация и модель составляют репозиторий документации. Документация хранится в виде набора XML—файлов. Модель также сериализуется в XML, таким образом, репозиторий представляется в виде набора XML—файлов, хранящихся на диске в отдельном каталоге.

Менеджер публикации предоставляет возможность получить документы в итоговых форматах, таких как PDF, HTML, HTMLHelp и.т.п. Параметры публикации документации настраиваются при написании текстов, так что публикация документации может выполняться в пакетном режиме. Поскольку для форматирования документации используется язык DocBook, публикацию документации также можно осуществлять с использованием технологии DocBook: менеджер публикации осуществляет сборку документов в соответствии с данными, указанными в специализированном информационном продукте, генерирует документ в формате DocBook, затем с помощью инструментальных средств, входящих в поставку DocBook, осуществляется публикация итоговых документов.

## 2.5 Апробация

Представленный метод был апробирован в проекте разработки семейства систем автоматизации телевещания [1]. В ходе апробации автором был реализован прототип менеджера публикации, который собирал информацию из репозитория документации и из файлов Microsoft Visio и обеспечивал публикацию итоговых документов в форматах PDF, HTML, а также HTML Help. В результате использования метода был достигнут высокий уровень повторного использования текста, однако был вскрыта и возможная проблема – для применения метода требуется технический писатель, знакомый с технологией XML. В небольших компаниях такого человека может не быть. В данном проекте проблема была решена с помощью временного привлечения программиста, знакомого с XML. Облегчить задачу технического писателя поможет интеграция инструментальных средств с одним из популярных XML—редакторов. В ходе апробации потребовалось расширить пакет инструментальных средств приложением, которое в пакетном режиме во время публикации документации извлекало диаграммы из пакета Microsoft Visio и интегрировало их в конечную документацию. Такое расширение – пример

адаптации пакета инструментальных средств под особенности конкретного семейства программных продуктов.

## Заключение

Методы разработки семейств программных продуктов уверенно набирают популярность, поскольку обеспечивают преимущества, которые недоступны при разрозненной разработке ПО. Представленный в данной работе метод разработки документации семейств программных продуктов дополняет их и предоставляет возможность вариативного повторного использования текстов документации. Метод был апробирован в проекте разработки документации семейства систем автоматизации телевещания [1]. Несмотря на выявленные сложности, применение метода упростило задачу разработки и сопровождения документации. В ходе дальнейших исследований планируется устранить найденные недостатки, а также усилить поддержку вариативности повторного использования за счет распространения идеи адаптеров на случай мелкозернистого повторного использования. Также автор планирует обобщить метод на более широкий класс семейств документов.

## Список литературы


[1] Кознов Д.В., Перегудов А.Ф., Романовский К.Ю., Кашин А, Тимофеев А. Опыт использования UML при создании технической документации. // "Системное программирование, вып. 1". – 2005 . – с. 18—36.

[2] Atkinson C., Bayer J., Bunse C., et al. Component—Based Product Line Engineering with UML. – Addison-Wesley Professional, 2001. – 464 p.

[3] Bassett, P. Framing software reuse – lessons from real world. – Yourdon Press, Prentice Hall, 1997.

[4] Batory D., Lofaso B., Smaragdakis Y. JST: Tools for Implementing Domain—Specific Languages. // Proc. 5th Int. Conf. on Software Reuse, Victoria, BC, Canada. – 1988. – P. 143–153

[5] Clark D. Rhetoric of Present Single—Sourcing Methodologies. // SIGDOC'02, October 20–23, 2002, Toronto, Ontario, Canada. – 2002.- P. 20–25

[6] Clements, P., Northrop, L. A Framework for Software Product Line Practice. – Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. http://www.sei.cmu.edu/plp/framework.html, 2006.

[7] Clements, P., Northrop, L. Software Product Lines: Practices and Patterns. – Boston, MA: Addison-Wesley, 2002. – 608 p.

[8] Czarnecki K., Eisenecker U., Generative Programming: Methods, Tools, and Applications. – Reading, Mass.: Addison Wesley Longman, 2000. – 864 p.

[9] Day, D., Priestley, M., Schell, David A. Introduction to the Darwin Information Typing Architecture – Toward portable technical information. // http://www-106.ibm.com/developerworks/xml/library/x-dita1/

[10] Eclipse project official site. // http://www.eclipse.org

[11] Eriksson M., B?rstler J., Borg K. The PLUSS Approach – Domain Modeling with Features, Use Cases and Use Case Realizations. // Proceedings of Software Product Line Conference'2005. Lecture notes in computer science vol. 3714, 2005. – P. 33–44

[12] Greenfield J., Short K., Cook S., Kent S. Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. – Indianapolis, Indiana: Wiley Publishing, Inc., 2004. – P. 696

[13] Griss M., Favaro J., d' Alessandro M. Integrating Feature Modeling with RSEB. // IEEE Proceedings of Fifth International Conference on Software Reuse, 1998. – P. 76–85

[14] Harrison N.. The Darwin Information Typing Architecture (DITA): applications for globalization. // 2005 IEEE International Professional Communication Conference Proceedings, 2005.

[15] Jarzabek, S., Bassett, P., Hongyu Zhang, Weishan Zhang. XVCL: XML-based variant configuration language. // 25th International Conference on Software Engineering, 2003. Proceedings. 3-10 May 2003. – P. 810–811.

[16] Jarzabek S., Knauber P.. Synergy between Component—based and Generative Approaches. // Proc. ESEC/FSE'99 Joint 7th European Software Engineering Conference and 7th ACM SIGSOFT Symposium on the Foundations of Software Engineering Toulouse, France,

September 1999, Lecture Notes in Computer Science No. 1687, 1999. – P. 429–445

[17] Kang K., Cohen S., Hess J., Novak J., et al. Feature—Oriented Domain Analysis (FODA) Feasibility Study. // Technical Report, CMU/SEI—90—TR—21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1990

[18] Krueger C., Easing the Transition to Software Mass Customization. // Proc. 4th Int'l Workshop on Software Product Family Eng., Springer-Verlag. – 2001. – P. 282–293.

[19] Kyo C. Kang, Jaejoon Lee, Patrick Donohoe. Feature—Oriented Product Line Engineering. // IEEE Software July/August 2002. – P. 58–65.

[20] Lee K., Kang K., Lee J. Concepts and Guidelines of Feature Modeling for Product Line Software Engineering. // C. Gacek, ed., Proc. 7th Int'l Conf. Software Reuse, Springer Lecture Notes in Computer Science, vol. 2319, Heidelberg, Germany, 2002. – P. 62–77

[21] Marques. M. Single-sourcing with FrameMaker. // TECHWR-L Magazine Online, http://www.techwr-l.com/techwhirl/magazine/technical/singlesourcing.html

[22] Netron Fusion // http://www.netron.com/

[23] Tolvanen J., Kelly S. Defining Domain—Specific Modeling Languages to Automate Product Derivation: Collected Experiences. // Proceedings of Software Product Line Conference'2005. Lecture notes in computer science vol. 3714, 2005. – P. 198–209

[24] Walsh N., Muellner L. DocBook: The Definitive Guide. – O'Reilly, 2003.

[25] http://www.authorit.com/

[26] http://www.xml.org